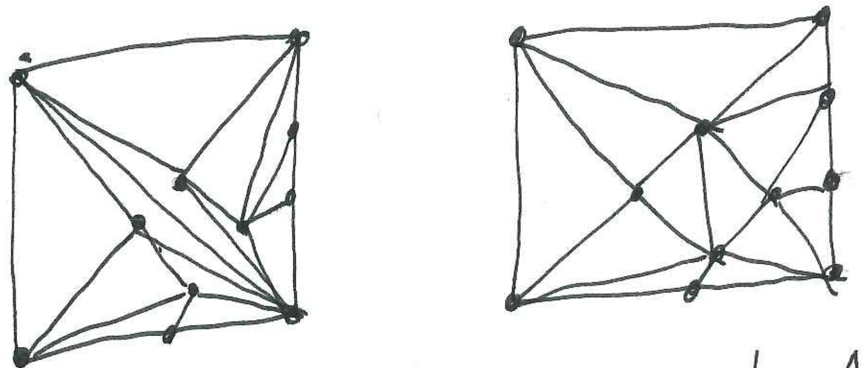
Why Delaunay?



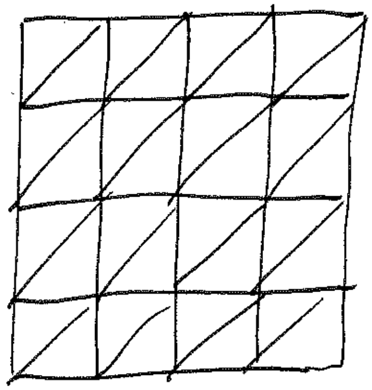
What makes a good Δ^n?

Intuition: (1) Should represent "local" geometry.

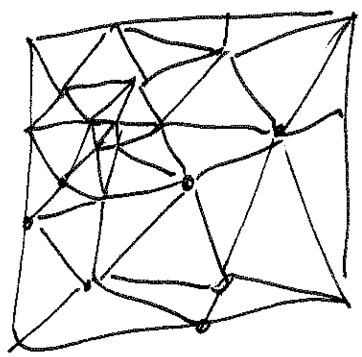
(2) Don't connect a to b if other points are in the way.

Empty Circle Property

Max-Min Angle Property also helps.



The Delaunay Δ^n is more "grid-like" for points that may not be on a grid.



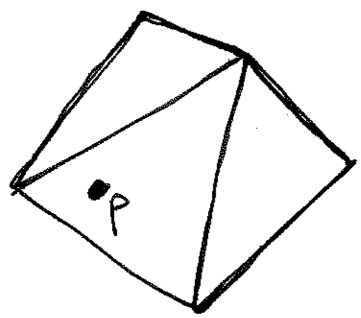
Randomized Incremental Delaunay Δ^n.

```
RIDel (P: point set)
  Randomly order the points of P
  Init Δⁿ T with an infinite Δ  ← requires special predicates.

  For i = 1 to n
  { [Find the triangle t in T containing p_i]  ← Point Location
    T ← T.split3ways(t)
    // Flip T to Delaunay
    push 3 edges of t onto work_queue.
    while (work_queue nonempty)
    { e = work_queue.pop.
      If e is not LD
       { T ← T.flip(e)
           push edges in new Δs to work queue
       }
    }
  }
  return T
```

Point Location for Randomized Incremental Delaunay Δ^n

The Problem: To insert a point, we need to know what triangle contains that point.

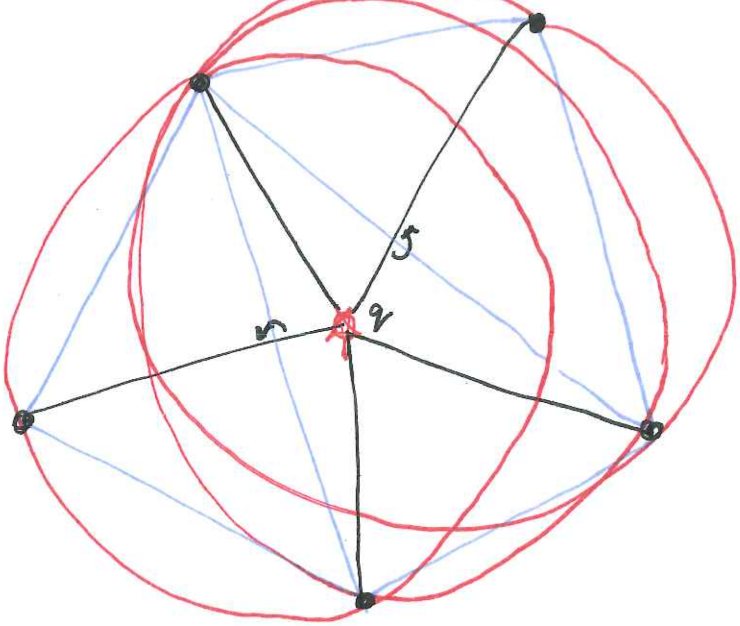


The idea: After adding $p_1, \dots, p_i$, we keep track of which points of $p_{i+1}, \dots, p_n$ are in which triangles of $Del_{\{p_1, \dots, p_i\}}$.

Then, when we insert a new point, we update for each triangle that was removed. Do this locally for each flip.

To Store: For each Δ, store a list of uninserted points it contains.

For each uninserted point, store the triangle that contains it.

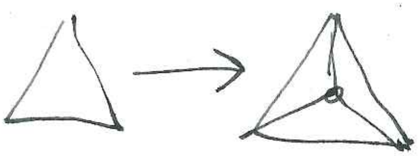


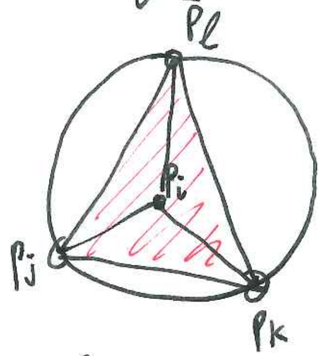
Given Del_P, we say q <u>encroaches</u> $\Delta \in Del_P$ if $q \in circle(\Delta)$.

If we insert q, all triangles that q encroaches will be destroyed.
All <u>new</u> triangles have a vertex at q, otherwise, they were already Delaunay.

Two ways to destroy a triangle.

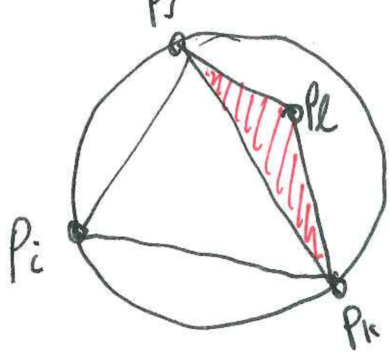
(1)





Both cases, p_i is inserted. $\Delta\, p_j p_k p_l$ <u>was</u> Del.

(2)



<u>PL work:</u>
any points that are moved on insert(p_i) are inside circle($p_j p_k p_l$)

Let $K(\Delta)$ be the points of P inside circle(Δ).

So, point location work (PL) is

$$O\left(\sum_{\Delta \in T} |K(\Delta)|\right)$$

Where T is the set of all triangles appearing in any of the Delaunay triangulations during the course of the algorithm.

$$Q_i = \{p_1, \dots, p_i\}$$

T_r is the set of triangles of Del_{Q_r}

$T_r \setminus T_{r-1}$ is the set of "new" triangles added when we added p_r.

$$k_r(q) = \left|\{\Delta \in T_r : q \in K(\Delta)\}\right|$$ ← # of triangles in Del_{Q_r} that q encroaches

$$k_r'(q) = \left|\{\Delta \in T_r \setminus T_{r-1} : q \in K(\Delta)\}\right|$$ ← # of new triangles that q encroaches

(1) $$\sum_{\Delta \in T_r \setminus T_{r-1}} |K(\Delta)| = \sum_{q \in P \setminus Q_r} k_r'(q)$$ ← standard double counting argument

(2) $$E\left[k_r'(q)\right] \leq \frac{3}{r} k_r(q)$$ because each triangle has $\frac{3}{r}$ chance of being "new".

$$E\left[\sum_{q \in P \setminus Q_r} k_r'(q)\right] \leq E\left[\sum_{q \in P \setminus Q_r} \frac{3}{r} k_r(q)\right]$$

(3) $$E\left[k_r(p_{r+1})\right] = \frac{1}{n-r} E\left[\sum_{q \in P \setminus Q_r} k_r(q)\right]$$ because each of the $n-r$ uninserted points are equally likely to be chosen as p_{r+1}.

(4) $$k_r(p_{r+1}) = |T_r \setminus T_{r+1}|$$ It's the # of Δs destroyed.

$$= |T_{r+1} \setminus T_r| - 2$$ Which is 2 less than the number of new Δs.

(5) $$E\left[|T_{r+1} \setminus T_r|\right] < 6$$ By Euler's Formula

$$E\left[\sum_{\Delta \in T} |K(\Delta)|\right] = E\left[\sum_{r=1}^{n} \sum_{\Delta \in T_r \setminus T_{r-1}} |K(\Delta)|\right]$$ [by telescoping (partit]

$$= \sum_{r=1}^{n} \sum_{q \in P \setminus Q_r} E\left[k'_r(q)\right]$$ [by (1)]

$$\leq \sum_{r=1}^{n} E\left[\sum_{q \in P \setminus Q_r} \left(\frac{3}{r}\right) k_r(q)\right]$$ [by (2)]

$$= \sum_{r=1}^{n} \frac{3(n-r)}{r} E\left[k_r(p_{r+1})\right]$$ [by (3)]

$$= 3 \sum_{r=1}^{n} \left(\frac{n-r}{r}\right) E\left[|T_{r+1} \setminus T_r| - 2\right]$$ [by (4)]

$$< 12 \sum_{r=1}^{n} \left(\frac{n-r}{r}\right)$$ [by (5)]

$$= O(n \log n)$$